HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering

Jianing Sun*
Layer 6 AI
Toronto, Canada
jianing@layer6.ai

Zhaoyue Cheng*
Layer 6 AI
Toronto, Canada
joey@layer6.ai

Saba Zuberi
Layer 6 AI
Toronto, Canada
saba@layer6.ai

Felipe Pérez
Layer 6 AI
Toronto, Canada
felipe@layer6.ai

Maksims Volkovs
Layer 6 AI
Toronto, Canada
maks@layer6.ai

ABSTRACT

Hyperbolic spaces offer a rich setup to learn embeddings with superior properties that have been leveraged in areas such as computer vision, natural language processing and computational biology. Recently, several hyperbolic approaches have been proposed to learn robust representations for users and items in the recommendation setting. However, these approaches don't capture the higher order relationships that typically exist in the recommendation domain. Graph convolutional neural networks (GCNs) on the other hand excel at capturing higher order information by applying multiple levels of aggregation to local representations. In this paper we combine these frameworks in a novel way, by proposing a hyperbolic GCN model for collaborative filtering. We demonstrate that our model can be effectively learned with a margin ranking loss, and show that hyperbolic space has desirable properties under the rank margin setting. At test time, inference in our model is done using the hyperbolic distance which preserves the structure of the learned space. We conduct extensive empirical analysis on three public benchmarks and compare against a large set of baselines. Our approach achieves highly competitive results and outperforms leading baselines including the Euclidean GCN counterpart. We further study the properties of the learned hyperbolic embeddings and show that they offer meaningful insights into the data. Full code for this work is available here: https://github.com/layer6ai-labs/HGCF.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

Recommender Systems, Hyperbolic Embeddings, Graph Convolutions

*Authors contributed equally to this work.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450101

ACM Reference Format:
Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. 2021. HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3442381.3450101

1 INTRODUCTION

A central task in recommender systems is to accurately capture user preferences and item attributes to correctly predict whether a user will prefer a given item. Collaborative filtering (CF) approaches utilize past user-item interaction data to drive recommendations. Recent research has largely focused on the implicit feedback setting as implicit interactions can typically be collected in much larger quantity and at low cost. A prominent direction in CF is latent models that learn compact representations of users and items, distances between these representations are then used to infer preference. The most popular latent CF approach is matrix factorization where user-item interaction matrix is approximated by a product of two low-rank matrices that are taken as user and item representations. An alternative direction is to view the interaction matrix as a bipartite graph, with edges representing interactions between user and item nodes. Graph-based methods can then be applied to pass messages along the edges and learn representations that summarize neighbourhood information for each node.

A key advantage of the graph oriented approach is the explicit ability to model higher order relationships (neighbors-of-neighbors) between users and items. Recent work on applying graph convolution networks (GCNs) to CF has demonstrated the importance of exploring higher order relationships for user-item modeling [14, 36]. These approaches have achieved state-of-the-art performance on many public benchmarks by applying multiple levels of neighborhood aggregation under the graph convolutional setting to produce the final representations. Similarly strong performance from graph learning has been achieved in other domains such as computer vision [23, 38, 39], natural language processing [1, 19, 26] and computational biology [8, 10, 17].

In parallel, it has been shown that many real-world datasets exhibit the prototypical characteristics of complex networks such as the power-law degree distribution [31], including user-item graphs

in CF datasets [3]. These properties are related to the underlying hierarchical structure [31] which is well modelled by hyperbolic geometry [20]. This has motivated representation learning in hyperbolic space to more effectively capture the structure of the user-item graph. Proposed methods in this area have recently demonstrated that a considerable improvement in accuracy can be achieved by adapting variants of the traditional matrix factorization approach to the hyperbolic setting [3, 27, 35]. Furthermore, the learned hyperbolic representations were found to naturally capture both hierarchy and similarity through their norm and hyperbolic distance [24, 28].

Motivated by these advances, we propose a new hyperbolic GCN model for CF. We refer to our approach as the **H**yperbolic **G**raph **C**ollaborative **F**iltering (HGCF), and to the best of our knowledge this is the first successful combination of GCN and hyperbolic learning in recommender systems. HGCF learns user and item representations in the hyperbolic space by aggregating neighborhood information through a GCN module on the tangent space of a reference point. By taking advantage of the exponential neighborhood growth in the hyperbolic space, we show that our model can be effectively learned using a margin ranking loss based on hyperbolic distances optimized with the Riemannian Gradient Descent. We analyze margin learning under hyperbolic distance, and demonstrate that it has more capacity to distribute the items within a given margin than the equivalent Euclidean representation. Additional capacity can in turn alleviate the problem of dissimilar items being pushed closer together that was shown to affect margin learning in CF [22].

Extensive experiments on public benchmarks show that our approach outperforms many leading baselines including the Euclidean GCN counterpart, and is more robust to changes in embedding dimensionality. Further analysis of the item representations learned by our model reveals a natural hierarchical structure. In summary, our contributions are as follows:

- We propose a hyperbolic GCN architecture for CF and conduct an investigation into GCN layer designs to better facilitate learning in deeper models that explore higher order relationships.
- We analyze margin ranking optimization with hyperbolic distances and demonstrate that it has desirable properties for learning robust representations.
- We implement an optimization procedure for our model based on Riemannian Gradient Descent that propagates information through the GCN via the tangent space of a selected reference point.
- We conduct extensive experiments demonstrating superior performance and show that learned item representations capture meaningful structure in the interaction data.

2 RELATED WORK

In this section we review relevant previous work from the GCN and hyperbolic learning literature.

2.1 Graph Convolutional Neural Networks

GCN-based methods have received increasing attention due to their ability to learn rich node representations from arbitrarily structured graphs [19, 30]. They have been effectively applied in a wide range of domains such as computer vision [23, 38, 39], natural language processing [1, 19, 26] and computational biology [8, 10, 17]. In collaborative filtering, GCNs have been adapted for matrix factorization [36] and diversified recommendation [32], demonstrating leading performance. Subsequently, [14] empirically showed that the essential GCN component for CF is the iterative neighborhood aggregation. Other components such as feature transformation and nonlinear activations have little effect and can be removed. [33] proposes a neighbor interaction aware GCN approach that explicitly models the neighbor relationships in the user-item graph. [16] expands the GCN architecture to multi-behavior setting where several sources of user preference are available.

The majority of proposed GCN approaches embed nodes in the Euclidean space. Recently, [24] and [5] propose to learn GCNs in the hyperbolic space and show superior results on graph classification problems where graphs have hierarchical structure. We build on this work to bring the benefits of hyperbolic GCNs to the CF domain.

2.2 Hyperbolic Embedding Learning

Representation learning has taken an important role in extracting information out of semi-structured and unstructured data such as text or graphs. This type of data often contains an underlying hierarchical structure that is difficult to capture with representations in Euclidean space. To mitigate this problem [28] proposes learning representation in the Poincaré ball formulation of hyperbolic space that naturally captures hierarchical structure. Expanding on that work, [29] finds that learning representations based on the Lorentz formulation of the hyperbolic space is more efficient. Recently, hyperbolic representation learning has been applied to a variety of problems in different areas [4, 7, 12, 18], including CF [3, 9, 27, 35]. In CF, [27] uses a single layer autoencoder in hyperbolic space to learn user-item embeddings. [35] studies metric learning in hyperbolic space and its connection to CF. [3] proposes a weighted margin rank batch loss to learn a hyperbolic model and generates user representation by item aggregation in hyperbolic space via Einstein midpoint. Finally, [9] applies hyperbolic learning for point of interest recommendation. Our approach is related to these works in that we also learn user and item representations in hyperbolic space. However, a key difference is that our approach captures higher order information in user-item interactions by incorporating multiple levels of neighborhood aggregation through a GCN module.

3 PRELIMINARIES

Our aim is to learn d-dimensional user and item embeddings in the hyperbolic space. A number of equivalent mathematical formulations have been derived for the hyperbolic space. Here, we use the Lorentz formulation to define the model which is found to be more stable for numeric optimization [29]. We then use the Poincaré formulation to visualize the learned embeddings as it provides an intuitive way to lay out the points on the sphere. This property is particularly useful to visualize the hierarchical structure as distances stretch exponentially towards the sphere boundary. In this section we summarize the main properties of both formulations.

We recall that a d-dimensional hyperbolic space is a Riemannian manifold $\mathcal{M}$ with a constant negative curvature, which we denote

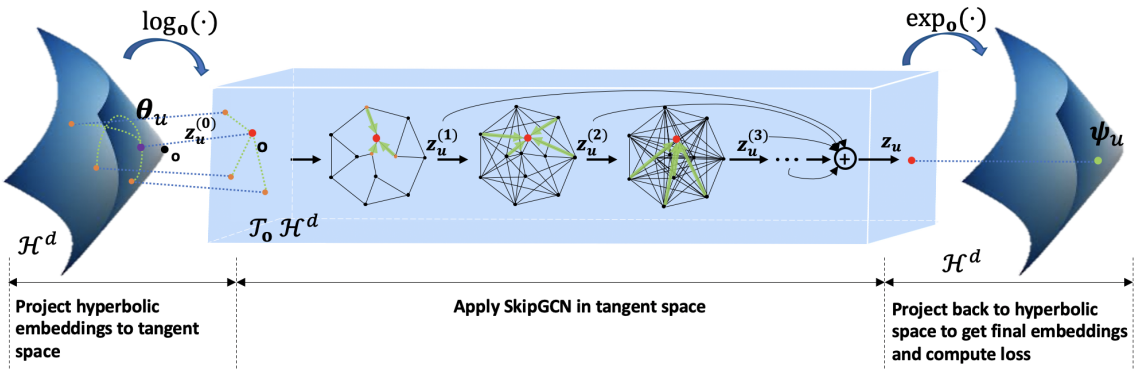


Figure 1: HGCF model architecture. Input user embedding $\boldsymbol{\theta}_u$ is projected into the tangent space $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ using $\mathbf{o}$ as the reference point in the hyperbolic space. The resulting representation $z_u^{(0)}$ is passed through multiple layers of SkipGCN to encode higher order neighborhood information. Output of SkipGCN z_u is then projected back to the hyperbolic space via an exponential map to get the final user embedding $\boldsymbol{\psi}_u$ which is used to compute the loss. Analogous derivation is done for item embeddings.

by c. Let $k = -1/c$ denote the negative reciprocal of the curvature such that $k > 0$. The tangent space $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ at point $\mathbf{x}$ on $\mathcal{M}$ is a d-dimensional Euclidean space that best approximates $\mathcal{M}$ around $\mathbf{x}$. The elements of $\mathcal{T}_{\mathbf{x}}\mathcal{M}$ are referred to as tangent vectors. The Lorentz and Poincaré formulations are equivalent mathematical representations of the hyperbolic space, and each is determined by an underlying set and a metric tensor. The Lorentz representation is defined by the pair $\mathcal{L}^d = (\mathcal{H}^d, g_{\mathcal{L}})$:

$$\mathcal{H}^d = \{\mathbf{x} \in \mathbb{R}^{d+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -k, x_0 > 0\} \tag{1}$$

where $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$ is the Lorentz inner product given by $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{d} x_i y_i$ for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{d+1}$, and the metric tensor is $g_{\mathcal{L}} = \text{diag}[-1, 1, 1, \cdots, 1]$. The metric $g_{\mathcal{L}}$ induces a distance function for any pair of points $\mathbf{x}, \mathbf{y} \in \mathcal{H}^d$ given by:

$$d_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \sqrt{k} \, \text{arcosh}\left(-\frac{\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}}{k}\right) \tag{2}$$

The tangent space centered at the point $\mathbf{x}$ in the Lorentz manifold is then defined by:

$$\mathcal{T}_{\mathbf{x}}\mathcal{H}^d = \{\mathbf{v} \in \mathbb{R}^{d+1} : \langle \mathbf{v}, \mathbf{x} \rangle_{\mathcal{L}} = 0\} \tag{3}$$

The Poincaré representation $\mathcal{B}^d$ has the open sphere $\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| < k\}$ as the underlying set, with curvature $c = -1/k < 0$ and where $\|.\|$ denotes the Euclidean norm. Distances in $\mathcal{B}^d$ are measured via the function :

$$d_{\mathcal{B}}(\mathbf{x}, \mathbf{y}) = \sqrt{k} \, \text{arcosh}\left(1 + 2k \frac{\|\mathbf{x} - \mathbf{y}\|^2}{(k - \|\mathbf{x}\|^2)(k - \|\mathbf{y}\|^2)}\right) \tag{4}$$

It is important to note here that for a fixed point $\mathbf{x}$ the distance $d_{\mathcal{B}}(\mathbf{x}, \mathbf{y})$ will increase exponentially towards infinity as $\mathbf{y}$ gets closer to the sphere boundary. To map between Lorentz and Poincaré formulations we apply the following transformation:

$$p_{\mathcal{L} \to \mathcal{B}}(\mathbf{x}) = p_{\mathcal{L} \to \mathcal{B}}(x_0, x_1, \ldots, x_d) = \sqrt{k} \frac{(x_1, \ldots, x_d)}{x_0 + \sqrt{k}} \tag{5}$$

4 OUR APPROACH

In this section we present our HGCF approach. We consider the standard implicit CF set-up with m users $U = \{u_1, .., u_m\}$ and n items $I = \{i_1, ..., i_n\}$. Interactions between users and items are given in a sparse $m \times n$ binary interaction matrix R where R_{ui} is 1 if user u interacted with item i and 0 otherwise. We denote $\mathcal{N}_u = \{i \in I : R_{ui} = 1\}$ as the set of items that user u interacted with and refer to this set as the user neighborhood. Similarly, the item neighborhood is given by $\mathcal{N}_i = \{u \in U : R_{ui} = 1\}$.

To apply HGCF, we first initialize embeddings for all users and items in the hyperbolic space. Then, given a user u with corresponding embeddings $\boldsymbol{\theta}_u \in \mathcal{H}^d$, we map $\boldsymbol{\theta}_u$ to the tangent space of a reference point where it is passed through several layers of graph convolutions. The updated embedding, with encoded neighbor information, is mapped back to $\mathcal{H}^d$ where a hyperbolic margin ranking loss is applied. The signal from the loss is back-propagated to update relevant parameters and the process is repeated. An analogous procedure is applied to item embeddings. Figure 1 summarizes the HGCF architecture and we describe each component in detail below.

4.1 Embeddings in Hyperbolic Space

We use the Lorentz representation for both user and item embeddings. We fix the origin $\mathbf{o} = (\sqrt{k}, 0, \cdots, 0) \in \mathcal{H}^d$ and use it as a reference point. Note that $k = -1/c$ is the reciprocal of the curvature c that is considered a hyper-parameter here and set empirically. The embeddings are initialized by sampling from the Gaussian distribution on the tangent space $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ of the reference point $\boldsymbol{o}$. Formally, given a user u and an item i, we first sample from the multivariate Gaussian:

$$\boldsymbol{\theta}'_u, \boldsymbol{\theta}'_i \sim N(\mathbf{0}, \sigma \boldsymbol{I}_{d \times d})$$

and then set:

$$\boldsymbol{\theta}''_u = [0; \boldsymbol{\theta}'_u] \qquad \boldsymbol{\theta}''_i = [0; \boldsymbol{\theta}'_i]$$

where $[;]$ denotes concatenation. Note that both $\boldsymbol{\theta}''_u$ and $\boldsymbol{\theta}''_i$ satisfy $\langle \boldsymbol{\theta}''_u, \boldsymbol{o} \rangle_{\mathcal{L}} = 0$ and $\langle \boldsymbol{\theta}''_i, \boldsymbol{o} \rangle_{\mathcal{L}} = 0$ and therefore belong to $\mathcal{T}_o\mathcal{H}^d$. To

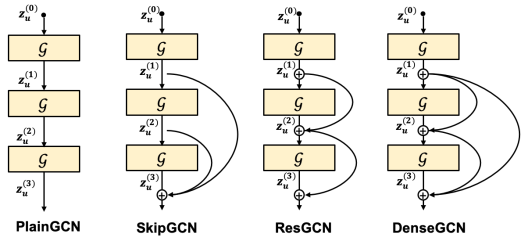


Figure 2: Graph convolution architectures, $\mathcal{G}$ represents the graph convolution layer. PlainGCN is the original GCN architecture, and the other three architectures have different structures of skip connections.

obtain the corresponding embeddings in $\mathcal{H}^d$ we project into the hyperbolic space via the exponential map $\exp_{\mathbf{o}} : \mathcal{T}_{\mathbf{o}}\mathcal{H}^d \rightarrow \mathcal{H}^d$, defined as [5]:

$$\exp_{\mathbf{o}}(\mathbf{v}) = \cosh\left(\frac{\|\mathbf{v}\|_{\mathcal{L}}}{\sqrt{k}}\right)\mathbf{o} + \sqrt{k}\sinh\left(\frac{\|\mathbf{v}\|_{\mathcal{L}}}{\sqrt{k}}\right)\frac{\mathbf{v}}{\|\mathbf{v}\|_{\mathcal{L}}} \tag{6}$$

where $\mathbf{v} \in \mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ and $\|\mathrm{v}\|_{\mathcal{L}} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v}\rangle_{\mathcal{L}}}$. We then get user and item embeddings in $\mathcal{H}^d$:

$$\boldsymbol{\theta}_u = \exp_{\mathbf{o}}(\boldsymbol{\theta}''_u) \qquad \boldsymbol{\theta}_i = \exp_{\mathbf{o}}(\boldsymbol{\theta}''_i) \tag{7}$$

These hyperbolic embeddings are used to initialize the model for all users $\{\boldsymbol{\theta}_u\}_{u\in U}$ and items $\{\boldsymbol{\theta}_i\}_{i\in I}$.

4.2 Skip-connected Graph Convolution Networks

The main idea behind GCNs is to learn node representations within a graph by iteratively aggregating local information from multi-hop neighbors. The aggregation process typically involves feature transformations and non-linear activation at each stage. However, recent work found that the gain from feature transformation and non-linearity is minimal over the simpler approach of mean aggregation [14]. Moreover, non-linearity adds considerable representational power to the model and can lead to significant over-fitting on highly sparse CF datasets [14]. In light of these findings, we opt to remove feature transforms and non-linearities to both reduce model complexity and speed up training and inference.

In hyperbolic space the analog of mean aggregation is the Fréchet mean which has no closed form solution [11]. To deal with this problem, we perform aggregation in the tangent space $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$. To this end we first project the embeddings $\boldsymbol{\theta}_u, \boldsymbol{\theta}_i$ to $\mathcal{T}_{\mathbf{o}}\mathcal{H}^d$ via the logarithmic map $\log_{\mathbf{o}} : \mathcal{H}^d \rightarrow \mathcal{T}_{\mathbf{o}}\mathcal{H}^d$. For the Lorentz representation this logarithmic map is defined as [5]:

$$\log_{\mathbf{o}}(\mathbf{x}) = \sqrt{k}\,\mathrm{arcosh}\left(-\frac{\langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}}{k}\right)\frac{\mathbf{x} + \frac{1}{k}\langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}\mathbf{o}}{\|\mathbf{x} + \frac{1}{k}\langle \mathbf{o}, \mathbf{x}\rangle_{\mathcal{L}}\mathbf{o}\|_{\mathcal{L}}} \tag{8}$$

where $\mathbf{x} \in \mathcal{H}^d$, $\mathbf{o} \in \mathcal{H}^d$ and $\mathbf{x} \neq \mathbf{o}$. The resulting vectors serve as input to the first GCN layer and we denote them as $\boldsymbol{z}_u^{(0)}$ and $\boldsymbol{z}_i^{(0)}$:

$$\boldsymbol{z}_u^{(0)} = \log_{\mathbf{o}}(\boldsymbol{\theta}_u) \qquad \boldsymbol{z}_i^{(0)} = \log_{\mathbf{o}}(\boldsymbol{\theta}_i) \tag{9}$$

Given the user and item neighborhoods $\mathcal{N}_u$ and $\mathcal{N}_i$, each graph convolutional layer is computed by aggregating neighborhood representations from the previous layer:

$$\boldsymbol{z}_u^{(l+1)} = \boldsymbol{z}_u^{(l)} + \sum_{i\in N_u}\frac{1}{|\mathcal{N}_u|}\boldsymbol{z}_i^{(l)} \qquad \boldsymbol{z}_i^{(l+1)} = \boldsymbol{z}_i^{(l)} + \sum_{u\in N_i}\frac{1}{|\mathcal{N}_i|}\boldsymbol{z}_u^{(l)} \tag{10}$$

We apply normalization by degrees $|\mathcal{N}_u|$ and $|\mathcal{N}_i|$ to ensure that the scale of embeddings does not increase with the number of layers. Each layer aggregates over increasingly higher order neighbors allowing to explicitly model long range relationships between users and items.

To fully exploit higher order relations we need to stack multiple graph convolutional layers together. However, previous work has found that stacking layers often results in significant drop in performance (even after several layers) due to gradient vanishing or over-smoothing [40]. To mitigate the depth limitation we explore architectures that contain skip connections motivated by the residual networks [13] and related work on deep GCNs [21]. The architectures that we consider are shown in Figure 2. Here, PlainGCN is the original model and SkipGCN, ResGCN, and DenseGCN have different structures of skip connections. SkipGCN contains skip connections from each layer to the final layer, ResGCN has residual connections between consecutive layers and DenseGCN combines SkipGCN and ResGCN. Empirically we find that SkipGCN performs the best in the hyperbolic setting and adopt this architecture for HGCF. Empirical comparison of these architectures is shown in Section 5.3.

Under SkipGCN the last layer aggregates representations from all intermediate layers and we get $\mathbf{z}_u = \mathbf{z}_u^{(L)} + \mathbf{z}_u^{(L-1)} + \ldots + \mathbf{z}_u^{(1)}$, and $\mathbf{z}_i = \mathbf{z}_i^{(L)} + \mathbf{z}_i^{(L-1)} + \ldots + \mathbf{z}_i^{(1)}$, where L is the total number of layers. The final embedding for each user and item is obtained by applying the exponential map in Equation 6 to project the output of SkipGCN back into the hyperbolic space:

$$\boldsymbol{\psi}_u = \exp_{\mathbf{o}}(\boldsymbol{z}_u) \qquad \boldsymbol{\psi}_i = \exp_{\mathbf{o}}(\boldsymbol{z}_i) \tag{11}$$

The updated embeddings now encode rich neighbor information, and we use the hyperbolic distance $d_{\mathcal{L}}(\boldsymbol{\psi}_u, \boldsymbol{\psi}_i)$ to estimate similarity between user-item pairs.

4.3 Hyperbolic Margin Ranking Loss

The margin ranking loss has proven to be highly effective for distance-based recommender models [15, 34]. This loss aims to separate positive and negative user-item pairs up to a given margin. Once the margin is reached the pairs are considered well separated and no longer contribute to the loss. This enables optimization to continuously re-focus on difficult pairs that violate the margin. We adopt this loss in our model and optimize for margin separation in the hyperbolic space. Formally, for each user u we sample a positive item i with $R_{ui} = 1$ and a negative item j with $R_{uj} = 0$, the goal is to separate i from j by their distance to u:

$$\mathrm{L}(u, i, j) = \max\left(d_{\mathcal{L}}(\boldsymbol{\psi}_u, \boldsymbol{\psi}_i)^2 - d_{\mathcal{L}}(\boldsymbol{\psi}_u, \boldsymbol{\psi}_j)^2 + m, 0\right) \tag{12}$$

where $d_{\mathcal{L}}$ is the hyperbolic distance (see Equation 2) and m is the margin. Note that once the difference $d_{\mathcal{L}}(\boldsymbol{\psi}_u, \boldsymbol{\psi}_i)^2 - d_{\mathcal{L}}(\boldsymbol{\psi}_u, \boldsymbol{\psi}_j)^2$

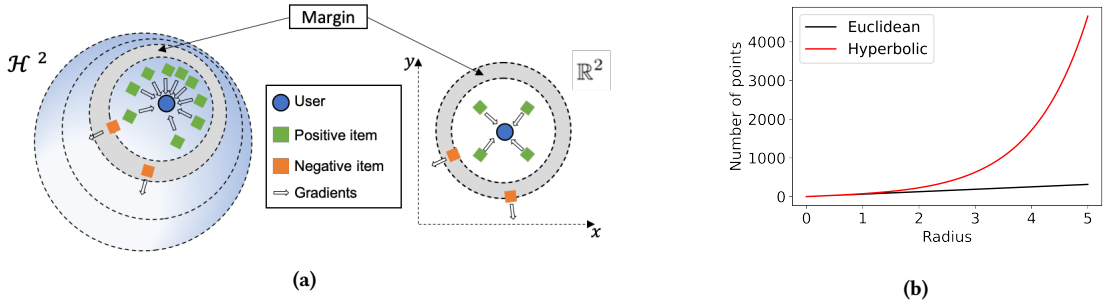


Figure 3: Margin ranking loss analysis in hyperbolic and Euclidean spaces. (a) Positive items are pushed closer to user while negative items are pushed outside the margin. Items in the hyperbolic space can be more concentrated while maintaining desired separation from each other since distances grow exponentially as we get closer to the sphere boundary. (b) Number of points that can be placed in two dimensional hyperbolic and Euclidean spaces at a given radius r from the user while maintaining a distance of at least $s = 0.1$ between each other.

is larger than the margin the loss becomes 0 and the item pair stops contributing to the gradient.

Despite strong performance, recent work has found that margin ranking loss with Euclidean distance can have a drawback where dissimilar items get pushed together [22]. One possible explanation for this phenomenon is that there isn't enough capacity to lay out the items so that they are both a minimum distance away from the user and from each other. We can analyze the number of items that can be placed at a fixed distance r from the user while also maintaining a separation of at least s from each other. In the Euclidean space this is bounded by the growth of the hypersurface $\{\boldsymbol{x} \in \mathbb{R}^d : d(\boldsymbol{x}, \boldsymbol{u}) = r\}$, which is a polynomial on r of degree $d-1$. In the hyperbolic space, the growth of the space $\{\boldsymbol{x} \in \mathcal{H}^d : d_{\mathcal{L}}(\boldsymbol{u}, \boldsymbol{x}) = r\}$ is exponential on r. For example, when $d = 2$ the number of points is $\left\lfloor \frac{2\pi r}{s} \right\rfloor$ in the Euclidean space and $\left\lfloor \frac{2\pi \sinh(r)}{s} \right\rfloor$ in the hyperbolic space with curvature $c = -1$. This is illustrated in Figure 3 where we contrast the Poincaré hyperbolic space with its Euclidean analog in two dimensions. Points in the hyperbolic space can be significantly more concentrated while maintaining the desired separation since distances grow exponentially near the sphere boundary. Our analysis indicates that margin ranking learning in the hyperbolic space has more solutions that adequately separate the items and avoid the undesirable collapse. This property can be a significant advantage particularly for large scale CF problems that have many users and items many of which are similar.

4.4 Model Training and Inference

Our aim is to optimize the user and item embeddings $\{\boldsymbol{\theta}_u\}_{u \in U}$ and $\{\boldsymbol{\theta}_i\}_{i \in I}$, and we apply the Riemannian SGD (RSGD) [2, 29, 37] to learn these parameters. RSGD mimics the stochastic gradient descent optimization while taking into account the geometry of the hyperbolic manifold. Following [29] and [37], given a user embedding $\boldsymbol{\theta}_u^{(t)}$ at iteration t, our optimization procedure consists of the following steps:

(1) Compute the gradient of the loss in Euclidean space $\nabla \mathrm{L} = \partial L / \partial \boldsymbol{\theta}_u$, $\nabla \mathrm{L} \in \mathbb{R}^{d+1}$. This gradient is computed by first finding the partial derivative $\partial L / \partial \boldsymbol{\psi}_u$ and then applying the chain rule to back-propagate it through the SkipGCN.

(2) Compute the Riemannian gradient $\nabla^{\mathcal{H}^d} \mathrm{L}$ by first computing $\boldsymbol{h}^{(t)} = g_{\mathcal{L}}^{-1} \nabla \mathrm{L}$ and then projecting $\boldsymbol{h}^{(t)}$ onto $\mathcal{T}_{\boldsymbol{\theta}_u^{(t)}} \mathcal{H}^n$:

$$\nabla^{\mathcal{H}^d} \mathrm{L} = \boldsymbol{h}^{(t)} + \frac{\langle \boldsymbol{\theta}_u^{(t)}, \boldsymbol{h}^{(t)} \rangle_{\mathcal{L}}}{k} \boldsymbol{\theta}_u^{(t)}$$

Here, $\nabla^{\mathcal{H}^d} \mathrm{L}$ is a vector on the tangent space of $\mathcal{H}^d$ that gives the direction of steepest descent.

(3) Estimate the update step using the exponential map and update the embedding with step size η (learning rate):

$$\boldsymbol{\theta}_u^{(t+1)} = \exp_{\boldsymbol{\theta}_u^{(t)}} \left(-\eta \nabla^{\mathcal{H}^d} \mathrm{L} \right)$$

To prevent overfitting we additionally apply weight decay during the update controlled by the hyper-parameter λ.

Once the model is trained we run run inference to recommend new items to each user. We compute the squared hyperbolic distance on $\mathcal{H}^d$ between user u and every item in the dataset that is not in the training set. We then select top items with the smallest distance and output them as recommendations for u.

Table 1: Dataset statistics.

Dataset	#User	#Item	#Interactions	Density
Amazon-CDs	22,947	18,395	422,301	0.100%
Amazon-Books	52,406	41,264	1,861,118	0.086%
Yelp2020	91,174	45,063	1,940,014	0.047%

5 EXPERIMENTS

We conduct extensive experiments on public datasets and benchmark HGCF against leading CF baselines. We use the Amazon-CDs, Amazon-Books and Yelp2020 datasets, and their statistics are summarized in Table 1. These datasets vary in size and sparsity providing a robust performance measure. Each dataset is split into the 80-20 training and test sets, and all models are evaluated on the test set using Recall and NDCG metrics. Following previous work, we convert the ratings into binary preferences by applying a threshold ≥ 4 which simulates the implicit feedback setting [15, 25, 36].

Table 2: Recall (top table) and NDCG (bottom table) results for all datasets. The best performing model on each dataset and metric is highlighted in bold, and second best model is underlined. Asterisks denote statistically significant Wilcoxon signed rank test for the difference in scores between the best and second best models.

Datasets		BPRMF	WRMF	VAE-CF	TransCF	CML	LRML	SML	NGCF	LightGCN	HAE	HVAE	Ours
Amazon-CD	R@10	0.0779	0.0863	0.0786	0.0518	0.0864	0.0502	0.0475	0.0758	0.0929	0.0666	0.0781	**0.0962***
	R@20	0.1200	0.1313	0.1155	0.0791	0.1341	0.0771	0.0734	0.1150	0.1404	0.0963	0.1147	**0.1455***
Amazon-Book	R@10	0.0611	0.0623	0.0740	0.0407	0.0665	0.0522	0.0479	0.0658	0.0799	0.0634	0.0774	**0.0867***
	R@20	0.0974	0.0919	0.1066	0.0632	0.1023	0.0834	0.0768	0.1050	0.1248	0.0912	0.1125	**0.1318***
Yelp2020	R@10	0.0325	0.0470	0.0429	0.0247	0.0363	0.0326	0.0319	0.0458	0.0522	0.0360	0.0421	**0.0543***
	R@20	0.0556	0.0793	0.0706	0.0424	0.0638	0.0562	0.0544	0.0764	0.0866	0.0588	0.0691	**0.0884***

Datasets		BPRMF	WRMF	VAE-CF	TransCF	CML	LRML	SML	NGCF	LightGCN	HAE	HVAE	Ours
Amazon-CD	N@10	0.0610	0.0651	0.0615	0.0396	0.0639	0.0405	0.0361	0.0591	0.0726	0.0565	0.0629	**0.0751***
	N@20	0.0974	0.0817	0.0752	0.0488	0.0813	0.0492	0.0456	0.0718	0.0881	0.0657	0.0749	**0.0909***
Amazon-Book	N@10	0.0594	0.0563	0.0716	0.0392	0.0624	0.0515	0.0422	0.0655	0.0780	0.0709	0.0778	**0.0869***
	N@20	0.0971	0.0730	0.0878	0.0474	0.0808	0.0626	0.0550	0.0791	0.0938	0.0789	0.0901	**0.1022***
Yelp2020	N@10	0.0283	0.0372	0.0353	0.0214	0.0310	0.0287	0.0255	0.0405	**0.0461**	0.0331	0.0371	0.0458
	N@20	0.0512	0.0506	0.0469	0.0277	0.0428	0.0369	0.0347	0.0513	0.0582	0.0409	0.0465	**0.0585***

Table 3: Ablation analysis on the Amazon-CD and Amazon-Book datasets. LM is the latent model in Euclidean space optimized with margin ranking loss. LM+H is the hyperbolic version of LM, and LM+GCN adds SkipGCN to LM in Euclidean space. Our HGCF model is LM+GCN+H. Training and inference (top-k item retrieval) times are also shown for each model.

Arch.	Amazon-CD				Amazon-Book	
	R@10	R@20	Train (sec/epoch)	Inference (ms/user)	R@10	R@20
LM	0.054	0.086	0.37 ± 0.03	7.9 ± 1	0.041	0.069
LM+H	0.061	0.096	0.49 ± 0.06	11.3 ± 1	0.045	0.074
LM+GCN	0.086	0.131	1.04 ± 0.08	7.9 ± 1	0.077	0.117
HGCF	**0.096**	**0.145**	1.21 ± 0.02	11.3 ± 1	**0.087**	**0.132**

Experimental Settings We set the embedding dimension to 50 and adopt the same negative item sampling strategy across all models to make comparison fair. For all methods we follow the suggested settings in authors' code repositories, and use grid search with cross validation to select the best combination. Regularisation weight has a significant effect on performance and is selected from the same set of {0.00001, 0.0001, 0.001, 0.01, 0.1, 1} for each model. For WRMF, we select the positive item weight from {1, 10, 100, 1000, 10000}. For VAE-CF, we select the corruption value from {0.2, 0.3, 0.4, 0.5}. For margin-base methods we sweep over margins in [0.01, 2] and select 1.9 for CML and 1 for TransCF, LRML and SML. Additionally, for SML the λ and γ parameters are set to their default values of 0.01 and 10 which we found to work well. For HAE and HVAE we set curvature to 0.005 and use one layer in the encoder and decoder. Finally, for NGCF and LightGCN we set the number of GCN layers to 3 after evaluating architectures with 1-4 layers.

For our model, we set learning rate to 0.001, curvature $c = -1$ and use 3 graph convolution layers in SkipGCN for all datasets. Weight decay and margin are chosen separately for each dataset from the same base set as the baselines. All experiments are conducted with PyTorch on a server with 40 Intel Xeon CPU@2.20GHz cores and Nvidia Titan V GPU.

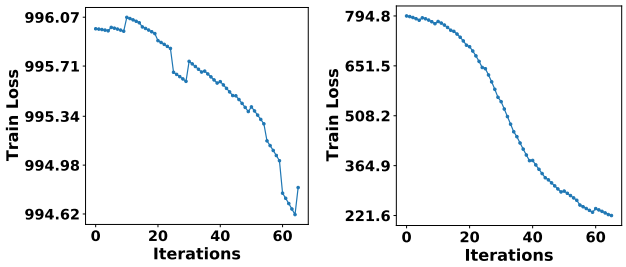
Table 4: Amazon-CD Recall@20 results for different GCN architectures (see Figure 2) as number of layers is varied from 1 to 4. All models are trained in hyperbolic space.

Number of layers	PlainGCN	SkipGCN	ResGCN	DenseGCN
1	**0.1418**	0.1418	0.1418	0.1418
2	0.1134	0.1440	**0.1440**	**0.1440**
3	0.0795	0.1455	0.1433	0.1265
4	0.0511	**0.1457**	0.1291	0.0980

5.1 Results

The results on all datasets are presented in Table 2. We see that HGCF consistently outperforms all baselines on all datasets and metrics except NDCG@10 on Yelp2020. The performance improvement relative to the best baseline LightGCN is larger on the more dense Amazon datasets. In particular, HGCF outperforms LightGCN by **8.5%** in Recall@10 and **11.4%** in NDCG@10 on the Amazon-Book dataset. These results indicate that hyperbolic representation can be effectively combined with GCNs in the recommendation setting, and can outperform state-of-the-art Euclidean GCN models.

We also see that our model outperforms the other hyperbolic baselines HAE and HVAE. Although both HAE and HVAE also learn hyperbolic use-item representations, they do so through autoencoders that take the entire interaction matrix as input and apply dense hyperbolic layers to encode and decode. This introduces significant additional training parameters that can make these models difficult to optimize on large datasets. Moreover, the autoencoder is optimized with the cross-entropy loss that treats each user-item pair separately, and can thus be a poor proxy for the target metrics such as NDCG that are ranking based. In contrast, our approach has no additional parameters beyond the user and item embeddings, and uses margin rank learning that optimizes the hyperbolic embeddings for relative order.



(a) PlainGCN **(b)** SkipGCN

Figure 4: PlainGCN and SkipGCN training curves for the first 60 iterations. Y-axis shows the total margin ranking loss at the end of each iteration.

5.2 Ablation Analysis

To asses the contribution from each component of HGCF to the overall performance we conduct an extensive ablation study with results shown in Table 3. Here, LM is the latent model that learns user and item embeddings with margin ranking loss in Euclidean space. LM+H is a hyperbolic version of LM where all embeddings are in hyperbolic space and are optimized using hyperbolic margin ranking loss. LM+GCN adds the SkipGCN module to LM but all learning is still done in Euclidean space. Finally, our model is LM+GCN+H that has both the SkipGCN module and hyperbolic representations. For each model we also show training and inference run time. Inference time denotes the time to retrieve top items for each user. Note that before inference we make forward passes through the GCN and cache the final user and item representations.

From the results in Table 3, we see that LM+H performs better than LM indicating that margin learning in hyperbolic space leads to better representations. This can be potentially attributed to the additional capacity that the hyperbolic space provides for margin separation towards the sphere boundary (as discussed in Section 4.3). We also see that adding the SkipGCN module to LM leads to a very significant improvement. This is consistent with previous work which shows that capturing higher order relationships through GCN is highly beneficial for CF [14, 36]. Finally, combining LM+GCN with hyperbolic learning produces the best results and further supports the conclusion that hyperbolic learning can be effectively combined with GCNs to maximize performance from both components.

From runtime results we see that, as expected, each additional component adds computational overhead during training. The most expensive part is the GCN module which computes multiple levels of embedding aggregation. Relative to that, the overhead from hyperbolic optimization is relatively minor. Overall, our HGCF model is approximately 3x more expensive to train than the LM baseline, but nearly doubles Recall@10 on the Amazon-CD and more than doubles it on the Amazon-Book datasets. During inference the GCN computation is cached so the difference is only in computing the hyperbolic distance vs dot product or squared distance in the Euclidean space. We see that the hyperbolic distance is around 40% more expensive than the Euclidean equivalent and inference increases from 7.9ms to 11.3ms per user.

Table 5: Recall@20 on Amazon-CD and Amazon-Book datasets as embedding dimensionality d is varied from 50 to 20.

Model	Amazon-CD			Amazon-Book		
	d=50	d=30	d=20	d=50	d=30	d=20
LightGCN	0.1406	0.1271	0.1155	0.1248	0.1074	0.0938
HGCF	0.1455	0.1360	0.1233	0.1318	0.1184	0.1042
Rel. Improv.	**3.48%**	**7.00%**	**6.75%**	**5.60%**	**10.24%**	**11.08%**

5.3 Comparison of GCN Architectures

We evaluate the effect that different residual and skip connection GCN architectures have on model performance. Table 4 shows Amazon-CD Recall@20 results for the architectures introduced in Section 4.2. We see that PlainGCN suffers from significant drop in performance even after 2 layers. This is consistent with previous work which also observed that stacking GCN layers quickly degrades performance [6, 40]. We note here that previous analysis was done in Euclidean space and our results show that this problem also persists in hyperbolic space.

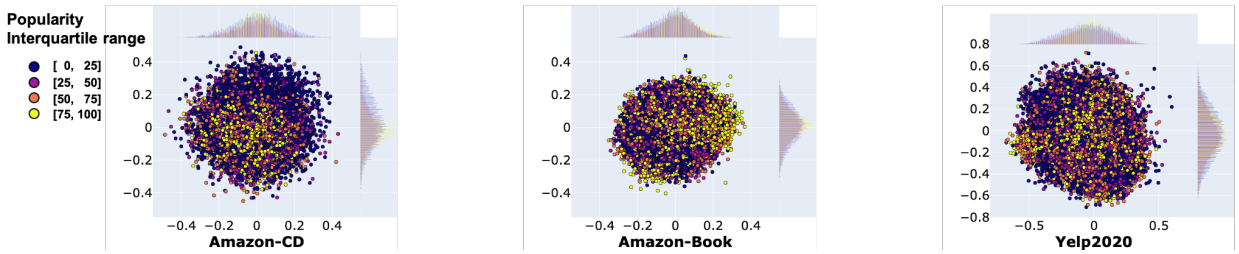
We additionally see that architectures with skip/residual connections perform better. Up to the first two layers these architectures are all identical and don't suffer from the same drop in performance as PlainGCN. However, after 2 layers SkipGCN continues to improve whereas ResGCN and DenseGCN start to drop in performance, and this drop becomes worse as more layers are added. We hypothesize that this happens because residual connections lead to significant over-smoothing making some representations indistinguishable. These results indicate that skip connections directly to the last layer provide a simple way to deal with optimization problems in deeper models, and work well in hyperbolic space. Figure 4 further shows the training curves for the 3-layer PlainGCN and SkipGCN models. The margin ranking loss barely decreases in PlainGCN after 60 interactions which further demonstrates the problem with deep GCN optimization. SkipGCN on the other hand can make swift progress with nearly 4x loss reduction.

5.4 Dimensionality Robustness

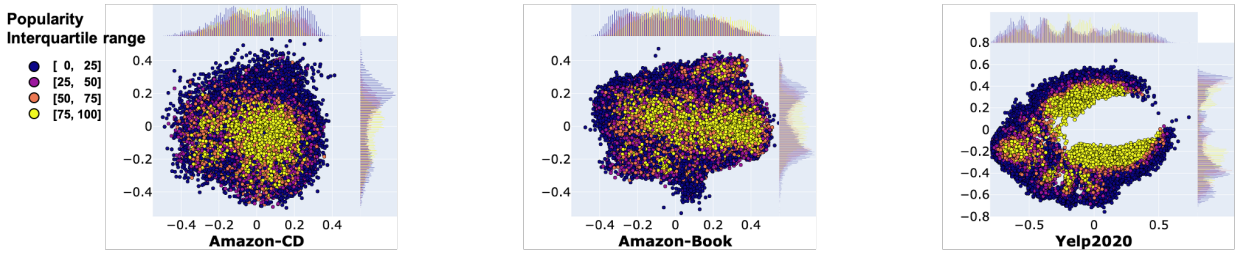
From our hyperbolic margin ranking loss analysis in Section 4.3, we concluded that hyperbolic space has more capacity to arrange points such that they satisfy distance margin requirements. This property is particularly important at lower dimensions where there are fewer degrees of freedom available. We evaluate this by reducing the embedding dimension d and comparing the performance of HGCF with the strongest Euclidean GCN baseline, LightGCN. The results are shown in Table 5. We observe that as embedding dimensionality is decreased from 50 to 20, both relative and absolute performance improvement of HGCF over LightGCN widen. In other words LightGCN suffers more from dimensionality reduction than HGCF. This reflects the advantage of hyperbolic geometry and the exponential expansion property of hyperbolic space [28]. Stronger performance with smaller embeddings can be highly relevant for large-scale recommender systems where both storage cost and retrieval speed are directly affected by embedding size.

5.5 Embedding Analysis and Visualization

The rich structure of hyperbolic space has been used to provide meaningful insights into the underlying data structure [28]. In this



(a) Item embeddings **before** the graph convolution layers.



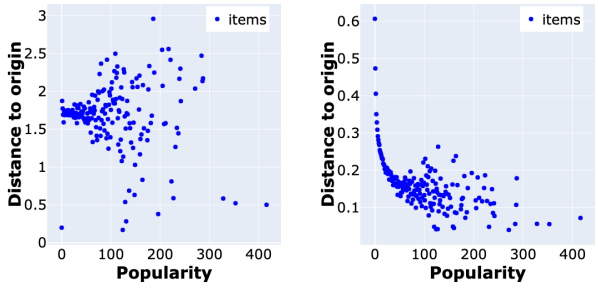
(b) Item embeddings **after** the graph convolution layers.

Figure 5: HGCF item embedding visualisation in the Poincaré representation of hyperbolic space before and after the SkipGCN graph convolutional layers. HGCF is trained with 4 graph convolutional layers and embedding dimension set to 2. Items are categorized into quartiles by popularity with [75, 100] representing the most popular items.

section we use the Poincaré formulation of hyperbolic space to visually analyse the embeddings learned by our model. We use a 4-layer HGCF model trained with embedding dimension set to 2 which allows to visualize the learned embeddings. To understand the effect of graph convolutions we plot item embeddings $\boldsymbol{\theta}_i$ and $\boldsymbol{\psi}_i$ before and after the SkipGCN module. The items are further segmented by the number of interactions (popularity) and we separately color each quartile.

Figure 5 shows these embedding plots for each of the three datasets. Before the graph convolution layers, items are organized in a circular region and are shuffled within that space. On the other hand, after the graph convolution layers, a clear hierarchy appears according to popularity. In the Amazon datasets this hierarchy is reflected in an almost circular way with the most popular items at the center and the less popular ones away from it. In the Yelp dataset, the embeddings are able to leverage the homogeneity of the hyperbolic space which leads to not only organization of items according to their popularity but also creation of different clusters of popular items.

We further visualize the differences between Euclidean and hyperbolic embeddings by plotting distance to the origin from each item vs its popularity. Figure 6 shows this plot for the Euclidean BPR and HGCF models trained on the Amazon-CD dataset with embedding dimension set to 2. We observe that BPR tends to cluster less popular items at specific distance from the origin. HGCF on the other hand has a clear exponential trend where distance to the origin increases exponentially for less popular items. This demonstrates that our model takes advantage of the exponentially growing volume in hyperbolic space and uses it to naturally cluster the items.



(a) BPR **(b)** HGCF

Figure 6: Distance from item embedding to origin (0, 0) vs item popularity for the Euclidean BPR model and HGCF. Both models are trained on the Amazon-CD dataset with embedding dimension set to 2.

6 CONCLUSION

In this paper, we propose a hyperbolic GCN collaborative filtering model HGCF. Each user and item embeddings is defined on hyperbolic space and then passed through multiple layers of skip-connected graph convolutions to encode higher order neighbor information. We adopt the margin ranking loss on the final embeddings produced by the GCN, and demonstrate that it has desirable separation properties under the hyperbolic learning setting. Experiments on three real-world datasets demonstrate that HGCF produces competitive performances compared to many state-of-the-art approaches and encodes meaningful structure in the learned representations. Future work involves further investigation into hyperbolic GCNs and in particular a direct way of modeling these architectures in the hyperbolic space.

REFERENCES

[1] Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[2] Silvere Bonnabel. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE Trans. Automat. Control* 58, 9 (2013).

[3] Benjamin Paul Chamberlain, Stephen R Hardwick, David R Wardrope, Fabon Dzogang, Fabio Daolio, and Saúl Vargas. 2019. Scalable hyperbolic recommender systems. *arXiv preprint arXiv:1902.08648* (2019).

[4] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[5] Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. In *Proceedings of the Neural Information Processing Systems*.

[6] Deli Chen, Yankai Lin, W. Li, Peng Li, J. Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[7] Bhuwan Dhingra, Christopher Shallue, Mohammad Norouzi, Andrew Dai, and George Dahl. 2018. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*.

[8] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the Neural Information Processing Systems*.

[9] Shanshan Feng, Lucas Vinh Tran, Gao Cong, Lisi Chen, Jing Li, and Fan Li. 2020. HME: A hyperbolic metric embedding approach for next-POI recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[10] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Proceedings of the Neural Information Processing Systems*.

[11] Maurice Fréchet. 1948. Les éléments aléatoires de nature quelconque dans un espace distancié. *Annales de l'institut Henri Poincaré* (1948).

[12] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. 2018. Hyperbolic attention networks. In *Proceedings of the International Conference on Learning Representations*.

[13] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016).

[14] X. He, K. Deng, X. Wang, Y. Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[15] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the International World Wide Web Conference*.

[16] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Multi-behavior recommendation with graph convolutional networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[17] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: Moving beyond fingerprints. (2016).

[18] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. 2020. Hyperbolic image embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[19] Thomas Kipf and M. Welling. 2017. Semi-supervised classification with graph convolutional networks. *Proceedings of the International Conference on Learning Representations* (2017).

[20] Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. 2010. Hyperbolic geometry of complex networks. *Physical Review E* 82, 3 (2010).

[21] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs go as deep as CNNs? *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019).

[22] Mingming Li, Shuai Zhang, Fuqing Zhu, Wanhui Qian, Liangjun Zang, Jizhong Han, and Songlin Hu. 2020. Symmetric metric learning with adaptive margin for recommendation.. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[23] Chundi Liu, Guangwei Yu, Maksims Volkovs, Cheng Chang, Himanshu Rai, Junwei Ma, and Satya Krishna Gorti. 2019. Guided similarity separation for image retrieval. In *Proceedings of the Neural Information Processing Systems Conference*.

[24] Qi Liu, Maximilian Nickel, and Douwe Kiela. 2019. Hyperbolic graph neural networks. In *Proceedings of the Neural Information Processing Systems*.

[25] Chen Ma, Liheng Ma, Yingxue Zhang, Ruiming Tang, Xue Liu, and Mark Coates. 2020. Probabilistic metric learning with adaptive margin for top-K Recommendation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

[26] Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[27] Leyla Mirvakhabova, Evgeny Frolov, Valentin Khrulkov, Ivan Oseledets, and Alexander Tuzhilin. 2020. Performance of hyperbolic geometry models on top-N recommendation tasks. In *Proceedings of the Recommender Systems Conference*.

[28] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. In *Proceedings of the Neural Information Processing Systems*.

[29] Maximilian Nickel and Douwe Kiela. 2018. Learning continuous hierarchies in the Lorentz model of hyperbolic geometry. In *Proceedings of the International Conference on Machine Learning*.

[30] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *Proceedings of the International Conference on Machine Learning*.

[31] Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *Physical review E* 67, 2 (2003), 026112.

[32] Jianing Sun, W. Guo, Dengcheng Zhang, Y. Zhang, Florence Regol, Y. Hu, H. Guo, Ruiming Tang, Han Yuan, X. He, and M. Coates. 2020. A framework for recommending accurate and diverse items using Bayesian graph convolutional neural networks. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

[33] Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, Chen Ma, and Mark Coates. 2020. Neighbor interaction aware graph convolution networks for recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[34] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *Proceedings of the International World Wide Web Conference*.

[35] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. HyperML: A boosting metric learning approach in hyperbolic space for recommender Systems.. In *Proceedings of the International Conference on Web Search and Data Mining*.

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[37] Benjamin Wilson and Matthias Leimeister. 2018. Gradient descent in hyperbolic space. *arXiv preprint arXiv:1805.08207* (2018).

[38] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. 2017. Scene graph generation by iterative message passing. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017).

[39] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. 2018. Graph R-CNN for scene graph generation. In *Proceedings of the European Conference on Computer Vision*.

[40] Lingxiao Zhao and Leman Akoglu. 2019. PairNorm: Tackling oversmoothing in GNNs. In *Proceedings of the International Conference on Learning Representations*.